Timing Diagnostics with PowerPC

Alternative scheme using TBR

Thu, Mar 15, 2001

The Digital PMC board (TechnoBox) includes a 32-bit 1MHz counter for diagnostic timing purposes. It also time-stamps clock events using this same counter. And for slower timing, a 2KHz counter is provided. It is expected that this will provide sufficient timing resources for use by the front-end software.

Another timing counter is available as part of the PowerPC chip architecture. This is the TBR, or Time Base Register. It counts in units of 66MHz cycles, although the exact rate depends upon the CPU board model. This timer can be used for any diagnostic timing, and it may be more quickly accessible than the registers on the DPMC board, which have slow access time.

The reason why the TBR is not used for all system timing needs is that we need to know time relative to clock event occurrences. The DPMC board decodes the clock and time-stamps each event automatically using its own microsecond counter. If we need to know times relative to clock events, we should use that same DPMC microsecond counter.

At this time, the use of careful hardware timing diagnostics reveals some apparent weakness in the DMPC board design. Use of TBM-based timing may help to diagnose these problems. This note is in part intended to design diagnostics to serve that purpose.

The timing of the interrupts delivered from the DMPC board is in question. It would be useful to know how consistently the 15 Hz cycle interrupts occur, and also the same details about the 40 ms timer interrupt, which serves as a backup in the absence of a 15 Hz clock event. (This interrupt also initiates Server Task processing.)

Histograms

Suppose we build a histogram of the elapsed times between successive 15 Hz cycle interrupts. This would have to be done by code added to that interrupt routine.

One can also build a histogram for the interrupt times of the 40 ms timer. This would have to be measured from the time of the software trigger that starts the 40 ms delay until the execution of the interrupt routine. This timer interrupt is not periodic; it is always started by software and the subsequent interrupt occurs just once.

Anomaly records

In addition to building histograms, one can also maintain a record of recent anomalous occurrences. Such a record can include the time-of-day along with the unusual value of delay detected. Separate records can be made for both the cycle and the 40 ms interrupts.

The form of the record could be the usual 16-byte data stream structure, which includes the time-of-day as the last 8 bytes. The first 8 bytes can include a type indicator, to identify which anomaly is being recorded, and the associated value of delay measured. The type can be a byte or a word. The delay value can be in microseconds as a 32-bit value. That leaves space for another word. The reason for trying to fit within the 16-byte range is for convenience in examining the data stream log in memory. One can examine it with the Memory Dump page, or one can print it out via the Print Memory page. One can also write a special page application to display and/or print such data in a more convenient form.

TBR logic

The value of the TBR is in two 32-bit parts, comprising a 64-bit value that rolls over after $2^{64}/(66*10^{6})$ seconds, or about 89000 years. It won't likely roll over at all, ever. There is a C language data type called "`long long`" that can hold a 64-bit value. We can keep a 64-bit value read from the TBR at the start of something to be timed, then reduce to a 32-bit value when computing the difference, or we can merely read the low 32 bits into a "`long`" variable. In these units, 32 bits amounts to 65 seconds, or 32 seconds if we consider "positive" differences only. This is far beyond anything needed for DPMC diagnostics. If convenient, it may be easier to keep differences in microsecond units. Software timing cannot be expected to be much better than that, anyway.

For histogramming interrupt times, perhaps bins of size 100 microseconds is good enough. This would require 66666/100 = 666 bins for measuring one 15 Hz cycle length. But suppose we had only millisecond bins. Then 66 bins or so would be needed. If we used 200 bins, that would be enough to measure two missing cycle interrupts. Perhaps 256 would be a convenient number of bins.

For an cycle interrupt anomaly log, anything departing from the usual 65–68 ms range might be considered noteworthy. The observed time could be reported in the log in microsecond units as a 32-bit value. The nominal 66.666 value is `0x0001046A`. For the 40 ms interrupt case, a value outside the range of 39–41 ms could be anomalous.

In both cases, it may be useful to record the most extreme value within the expected range, so that one has a handle on the usual range observed. Some of this variation in the cycle interrupt elapsed time can be caused by power line frequency fluctuation, which is expected.

Where does it go?

Where can the diagnostic data be stored? For a 256-bin array of 4-byte integer counts, we need 1K bytes. A diagnostic data stream log of 1K bytes would be enough for 64 entries, which may be enough. So, for diagnostics to monitor both the cycle and 40 ms interrupts, we might use 4K bytes, which can be in ordinary volatile memory.

Here is a suggested memory allocation strategy:

Address	*Size*	*Use*
`0000B000`	`400`	Cycle anomaly log
`0000B400`	`400`	Cycle histogram bins for 0–255 ms.
`0000B800`	`400`	40 ms anomaly log
`0000BC00`	`400`	40 ms histogram bins for 0–255 ms.

Data stream logs

The interrupt can probably call `DSWrite` to record into a data stream queue. But it may be more convenient to establish these logs independently. What is lost is being able to get listype support to the data stream, which would require a DSTRM entry, or maybe two. Since this data is only expected to be used ultimately to confirm that nothing anomalous is happening, or to compare between nodes that something funny happened with the clock event system, we may not need the formal attention granted it by installing DSTRM entries.

Log structure

The header can show the date when the diagnostics accumulation for both the histogram and the anomaly log began. Of course, the anomaly log can easily wrap, so we have only the most recent 64 (or 61, allowing for this header) anomalous records. If it is desired to maintain the lowest and highest values within the "OK" range, two 32-bit fields can be used for that purpose. The offset to the next anomaly log entry to be written must be maintained (16-bit). And the number of anomaly records ever written since the starting date should be kept (32-bits). For convenience, a copy of the last record written into the queue could be also kept in the header. A flag could be written to in order to cause a re-initialization of the diagnostics histogram and circular buffer.

The format of such a 48-byte header can be as follows:

Field	*Size*	*Meaning*
`inOffset`	2	Offset to next entry
`mxOffset`	2	Size of log structure
`totEnts`	4	Total number of records placed into queue
`baseTim`	4	Current reference time
`deltTim`	4	Current elapsed time
`validMin`	4	Minimum valid elapsed time deviation measured
`validMax`	4	Maximum valid elapsed time deviation measured
`initDate`	8	Time-of-day that histograms and logs were initialized.
`lastEnt`	16	Copy of last entry placed into log

The above 48-byte structure allows room for 61 records in 1K bytes.

By writing to the inOffset field an invalid value, such as 0000, the software (at task level?) might detect that this had happened and re-initialize the histogram and log.

The format of one 16-byte log record is as follows:

Field	*Size*	*Meaning*
recType	2	Type of this record
spare	2	(--)
delTime	4	Elapsed time in microseconds
dateTim	8	Time-of-day in usual BCD format

WIth the simple plan presented here, and with separate logs maintained for the two elapsed times being monitored, we may not need a recType field.